REMARKS

This Amendment responds to the Office Action mailed March 26, 2004 in the above-identified application. Based on the foregoing amendments and the following comments, reconsideration and allowance of the application are respectfully requested.

Claims 1-9 are pending in the application. Claims 1, 5 and 9 have been amended for the sole purpose of clarification and not to distinguish over the prior art of record. No new matter has been introduced. Claims 1, 5, 8 and 9 are independent claims.

The Examiner has rejected claims 1-7 and 9 under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. The Examiner asserts that the phrase "said at least some code sequences" in claims 1, 5 and 9 lacks antecedent basis. Claims 1, 5 and 9 have been amended to provide antecedent basis for the referenced language. The Examiner further asserts that the term "its" in claims 1, 5 and 9 is unclear. Claims 1, 5 and 9 have been amended for clarification. Accordingly, claims 1-7 and 9 are in full compliance with 35 U.S.C. §112, second paragraph, and withdrawal of the rejection is respectfully requested.

The Examiner has rejected claims 1-9 under 35 U.S.C. §103(a) as unpatentable over "Mac OS Runtime Architectures", 1997 publication (hereinafter Mac). The rejection is respectfully traversed.

Mac describes a PEF loader which includes relocations (Fig. 8-6 on page 8-15). Relocations are described beginning on page 8-21 of Mac.

Amended claim 1 is directed to a method of preparing an executable program from a plurality of object code modules, each module containing sets of section data and associated relocation instructions, and at least one of the modules further including a macro section containing code sequences, at least some of which are likely to be repeatedly included in the executable program and macro relocations associated with the macro section. At least one of the sets of section data includes at least one insertion location where at least some code sequences are to be inserted and wherein the associated relocation instructions include a macro call relocation identifying a location in the macro section. The method comprises, at link time when the executable program is prepared, reading the sets of section data and relocation instructions, on locating the macro call relocation, identifying the location in the macro section, and inserting

said at least some code sequences from the location in the macro section into the sets of section data at the insertion location. The at least some code sequences are selected by reading the macro relocations.

It is useful to clearly identify the context of the present application. The preamble of claim 1 recites "A method of preparing an executable program from a plurality of object modules". Mac contains no disclosure of such a method. The Examiner asserts that Mac teaches a method comprising a module represented by the PEF loader section shown in Fig. 8-6 on page 8-15 of Mac. However, Mac contains no disclosure that the PEF loader section comprises object code. More importantly, there is no disclosure in Mac of a plurality of object code modules (i.e. PEF loader sections) which are used to prepare an executable program. Thus, Mac does not disclose or suggest a method as defined by claim 1.

Claim 1 further requires that each module contains i) sets of section data and ii) associated relocation instructions. Also, claim 1 recites that at least one of said modules further includes a macro containing iii) code sequences and iv) macro relocations. Fig. 6 of the present application illustrates an example of this configuration by showing a single object code module 3 containing:

i) sets of section data as indicated by O1, O2 and O3;

ii) associated relocation instructions as indicated by R1, R2 and R3;

iii) code sequences as indicated by M1 and M2; and

iv) macro relocations as indicated by MR.

The Examiner contends that the loader section of Mac is analogous to the object code module of the present invention and that the routine "moo" is analogous to the code sequence of the macro section of the present invention. However, these analogies are clearly incorrect, since page 8-15 of Mac describes that the loader section is used to prepare a fragment and contains information imported and exported from the fragment. Moreover, Fig. 8-9 on page 8-25 of Mac clearly shows that the loader section is used to import the address of a routine which is found in another fragment. The last paragraph on page 8-24 states that at run time, all pointers to the routine "moo" in the calling fragment are set to zero, since the compiler cannot know the actual runtime address of the routine. Mac teaches how this is overcome using the loader section which

"fixes up" calling a fragment's pointer so that it points to the appropriate transition vector (address) for the routine "moo" in the called fragment.

The disclosure of Mac is different from the present invention for a number of reasons. In particular, claim 1 recites the feature of "inserting said at least some code sequences... into the set of section data at the insertion location". Mac contains no disclosure or suggestion of inserting the code sequence associated with the routine "moo" in the called fragment into the calling fragment. Instead, the correct address of the transition vector for the routine "moo" is merely fixed-up in the calling fragment but there is no disclosure of inserting any code sequences into the calling fragment. Moreover, this feature serves to illustrate a more fundamental distinction over Mac, in that the present application has an object code module containing i) sets of section data including at least one insertion location as well as iii) code sequences in the macro section, both of which appear in a single module. By contrast, in Mac not only are code sequences not inserted into the insertion location, as is the case in the present application, but additionally, Mac relies on a code sequence which is not contained in the same fragment, but rather its address needs to be imported from a called fragment.

Claim 1 is further distinguished over Mac in reciting "macro relocations associated with said macro section". The Examiner has not identified where such a feature can be found in Mac, and it is submitted that this feature is not taught by Mac. Moreover, claim 1 is distinguished in reciting that "said at least some code sequences being selected by reading the macro relocations". Claim 1 includes method steps for identifying the location in the macro section where at least some code sequences are to be found and inserting these from said locations into a set of section data at an insertion location and provides a further step of selecting which of said at least some code sequences are in fact inserted into the set of section data by reading the macro relocation. These steps are not disclosed or suggested by Mac.

Claim 1 is further distinguished over Mac in reciting "a macro section containing code sequences". This is clearly shown in Fig. 6 of the present application by reference numerals M1 and M2. In contrast, Mac discloses only a single routine "moo", i.e. only a single code sequence and therefore is not concerned with the problem of selecting which of said code sequences to select for the final executable program. In contrast, claim 1 requires selection of at least some code sequences. Thus, whereas the macro relocations of the present invention are used to select

which of the code sequences are in fact inserted into the executable program, Mac describes executing a relocation instruction at step 3 on page 8-26, which is simply used to supply the calling fragment with the correct address of the transition vector for the routine "moo".

The present invention is further distinguished over Mac in that claim 1 recites "at least some of which [code sequences] are likely to be repeatedly included in the execution program". The Examiner admits that this feature is not explicitly taught in Mac, but argues that on pages 8-13 and 8-14 Mac teaches replacing repeated code with small instructions that generate the same result. However, this has nothing to do with the present invention. This feature of Mac allows space to be saved in the PEF containers prior to execution. However, there is no intention of compressing the code sequences in the object module of the present application. Instead, this feature of claim 1 further emphasizes the distinct advantage of the macro section of the present application, which allows code sequences to be repeatedly included in the executable program.

For the reasons set forth above, amended claim 1 is clearly and patentably distinguished over Mac. Claims 2-4 depend from claim 1 and are patentable over Mac for at least the same reasons as claim 1.

Amended claim 5 is directed to a linker for preparing an executable program for a plurality of object code modules and contains limitations that are analogous to the limitations of claim 1. In particular, claim 5 requires, in part, that each object code module contains sets of section data and associated relocation instructions and further requires that at least one of the modules includes a "macro section containing code sequences" and "macro relocations associated with said macro section". These features are not disclosed or suggested by Mac. Claim 5 further requires a section data module "arranged to receive said at least some code sequences... to be inserted at the insertion location," and "a program preparing means which prepares said executable program including said set of section data with the inserted code sequences." As discussed above, Mac does not disclose or suggest preparing an executable program including said set of section data with the inserted code sequences. Instead, Mac describes a relocation instruction which supplies the calling fragment with the address of the transition vector for the called fragment. For these reasons and for the reasons discussed above in connection with claim 1, claim 5 is clearly patentable over Mac. Claims 6 and 7 depend from

claim 5 and are patentable over Mac for at least the reasons discussed above in connection with claims 1 and 5.

Claim 8 is directed to a method of assembling an object code module for linking to form an executable program. Claim 8 contains limitations that are analogous to the limitations of claim 1. In particular, claim 8 requires, in part, "naming a location in a macro section in the object code module containing a plurality of code sequences, at least some of which are likely to be repeatedly included in the executable program, marking at an insertion location in a set of section data in the object code module where at least some of said sequences are to be inserted", "generating in association with the section data a macro call relocation," and "generating a set of macro relocations associated with said macro section for selecting said at least some code sequences for insertion at the insertion location." As discussed above, Mac contains no disclosure or suggestion of inserting code sequences at an insertion location in the object code module and contains no disclosure or suggestion of selecting code sequences for insertion. For these reasons and for the reasons discussed above in connection with claims 1 and 5, claim 8 is clearly and patentably distinguished over Mac.

Amended claim 9 is directed to a computer program product in the form of an object code module and contains limitations that are analogous to the limitations of claim 1. The object code module contains "sets of section data and associated relocation instructions". The object code module further includes "a macro section containing code sequences" and "a set of macro relocations associated with said macro section." The computer program product is cooperable with a linker to cause execution of relocation operations in dependence on said relocations and including identifying the location in the macro section and "inserting said at least some code sequences from that location in the macro section in the set of section data at the insertion location." As discussed above in connection with claims 1, 5 and 8, Mac contains no disclosure or suggestion of an object code module including section data, associated relocation instructions, and a macro section containing code sequences and a set of macro relocations. Furthermore, Mac contains no disclosure of inserting code sequences in the section data at the insertion location, as required by amended claim 9. For these reasons and for the reasons discussed above in connection with claims 1, 5 and 8, amended claim 9 is clearly and patentably distinguished over Mac.

CONCLUSION

In view of the foregoing amendments and remarks, this application should now be in condition for allowance. A notice to this effect is respectfully requested. If the Examiner believes, after this amendment, that the application is not in condition for allowance, the Examiner is requested to call the Applicant's attorney at the telephone number listed below.

If this response is not considered timely filed and if a request for an extension of time is otherwise absent, Applicant hereby requests any necessary extension of time. If there is a fee occasioned by this response, including an extension fee, that is not covered by an enclosed check, please charge any deficiency to Deposit Account No. 23/2825.

Respectfully submitted,
Richard Shann, Applicant

By: William R. McClellan

William R. McClellan, Reg. No. 29,409
Wolf, Greenfield & Sacks, P.C.
600 Atlantic Avenue
Boston, Massachusetts 02210-2211
Telephone: (617) 720-3500

Docket No. S1022.80522US00
Date: June 23, 2004
x06/26/04x